# 41.7 LUA Grundlagen - EEPMain()

Autor:Goetz
Quelle: Mein EEP-Forum

---

Zeit aufzuräumen, bevor wir mit unseren Tests fortfahren.

Mit der Zeit wird es doch sehr unübersichtlich, wenn das Ausgabefenster alles behält, was man dort hinein geschrieben hat.

Zum Glück hat das Ausgabefenster einen Löschknopf. Drückt man auf den runden, blauen Knopf mit dem i darauf, dann findet man als letzten Punkt in der Liste "Lösche EEP Log"

Bequemer ist es natürlich, wenn Lua selbst aufräumt. Dafür gibt es in Lua eine fest definierte Funktion. Nämlich
**clearlog**()

Der Name dieser Funktion setzt sich aus zwei Worten zusammen:
*clear* ist das englische Wort für *klar*.
und *log* ist das englische Wort für ein *Protokoll*

Schreibt man an den Anfang des Skripts die Zeile
**clearlog**()
dann wird zuerst das Ausgabefenster geleert und dann alles andere ausgeführt.

Es gibt einige Funktionen, die in Lua fest definiert sind.
**clearlog**() ist eine davon. Die Funktion muss man nicht erst schreiben, weil sie in Lua schon enthalten ist.
**print**() ist ebenfalls eine dieser fest definierten Funktionen.

Aber die Funktion **EEPMain**() ist in Lua nicht definiert.
Die muss man selbst schreiben. Denn für jede Anlage wird eine andere, passende **EEPMain**() Funktion benötigt.

Vorsicht! Bitte nicht verwechseln!
**print**() ist in Lua definiert, muss aber im Skript aufgerufen werden um etwas zu machen.
**EEPMain**() ist nicht in Lua definiert, wird aber von EEP automatisch aufgerufen.

Und was muss nun in dieser Funktion **EEPMain**() drin stehen? Das untersuchen wir in diesem Kapitel genauer.

Ich würde vorschlagen, dass ihr wieder mit einer leeren Anlage beginnt. Dort öffnet ihr das Lua Skript Fenster und löscht alles, was dort drin steht. Komplett.

Dann schreibt ihr rein
**function EEPMain**()
und sonst nichts ...

Drückt auf "Skript neu laden". Ihr bekommt
**[string "EEP Script"]:1: 'end' expected near<eof>**
und das Skript Fenster bleibt geöffnet.

Frei übersetzt heißt das "Im EEP Skript wird vor dem Ende des Skripts ein 'end' erwartet.

Lua hat nämlich bei der Übertragung des Skripts geprüft, ob jede Funktion, die definiert wurde, auch ein Ende hat. Und unsere **EEPMain**() Funktion hat keins.

Wir müssen also mindestens schreiben:
```
function EEPMain()
end
```

Dann erneut auf "Skript neu laden" drücken und beobachten, was passiert:

Im Ausgabefenster steht noch die Fehlermeldung von zuvor und dann eine zweite, die lautet
**Error wrong result type #1**
was übersetzt heißt
Fehler: falsches Ergebnis Typ Nummer 1

Wer immer die Funktion **EEPMain**() aufgerufen hat, erwartet offenbar ein Ergebnis.

Funktionen können nämlich Daten bekommen und Daten wieder zurückgeben. Man könnte das vielleicht mit einem Verstärker vergleichen. Wie ein Verstärker haben auch Funktionen Ein- und Ausgänge.

Der Eingang einer Funktion sind die Klammern hinter dem Namen.

Und für den Ausgang gibt es den Befehl **return**

Bei **EEPMain**() interessiert nur der Ausgang. Im Kapitel "Funktionen" hatte ich schon einmal kurz angesprochen, dass Lua angehalten wird, wenn am Ende der Funktion **EEPMain**() das
```
return 1
```
fehlt. Das will ich jetzt etwas detaillierter erklären.

Probiert mal aus was passiert, wenn ihr stattdessen
```
return 2
```
schreibt. Im Ganzen also:

```
function EEPMain()
return 2
end
```

Löscht den Inhalt des Ausgabefensters, drückt auf "Skript neu laden", wechselt (wenn nötig) ins 3D Fenster und ihr seht im Ausgabefenster ... nichts.
Das ist jetzt nicht besonders aussagekräftig. Aber immerhin gibt es keine Fehlermeldung.

Besser wäre natürlich, wenn im Ausgabefenster etwas stehen würde was uns sagt, dass überhaupt etwas passiert.

Also erweitern wir das Skript ein wenig.
```
function EEPMain()
print("Lua rennt")
return 2
end
```

Ausgabefenster löschen, Skript neu laden und wenn man jetzt im 3D Modus ist, dann erscheint im Ausgabefenster
Lua rennt

Und zwar immer wieder. Zeile für Zeile untereinander.

Daraus kann man erkennen, dass die Funktion **EEPMain**() von EEP immer wieder aufgerufen wird.

Jetzt kann man bei **return** andere Zahlen probieren. Und stellt fest, dass sich dadurch nichts ändert. Nimmt man hingegen etwas anderes als eine Zahl und schreibt zum Beispiel:
**return** "nimm das"
dann bekommt man wieder die Fehlermeldung
**Error wrong result type #1**

Wenn **EEPMain**() von EEP aufgerufen wird, dann erwartet EEP, dass es eine Zahl zurück bekommt. Richtiger gesagt etwas vom Typ "Zahl". Aber was es in Lua und anderen Programmiersprachen mit diesen Typen auf sich hat, möchte ich an anderer Stelle erklären.

Welche Zahl man in **EEPMain**() zurück gibt, ist EEP egal.
Mit einer Ausnahme. Schreibt man
**return** 0
dann wird die Funktion **EEPMain**() nicht erneut aufgerufen.

Probiert es aus. Schreibt:
**function EEPMain**()
**print**("Lua rennt")
**return** 0
end

Löscht das Ausgabefenster, ladet das Skript neu, wechselt (wenn nötig) ins 3D Fenster und im Ausgabefenster steht
Lua rennt
genau ein einziges Mal. Mehr nicht. Es gibt also keine Fehlermeldung und die **EEPMain**() Funktion wird kein zweites Mal aufgerufen.

Man hat also bei der Implementierung der **EEPMain**() Funktion in EEP daran gedacht, dass man unter Umständen vielleicht den ständigen Aufruf stoppen möchte.

Solange Lua mit **return** eine Zahl zurück gibt, die nicht 0 ist, wird **EEPMain**() immer wieder aufgerufen. Damit entspricht die Funktion **EEPMain**() einem ständig kreisenden Schaltauto. Und was immer in der Funktion **EEPMain**() drin steht, wird bei jedem Durchlauf erneut ausgeführt.
Bis man mit **return** den Wert 0 zurück gibt. *Dann ist Schluss mit Lua* 

Ich empfehle einen sehr vorsichtigen Umgang mit der Funktion **EEPMain**(). Mit Schaltautos haben wir uns angewöhnt durch ständiges Kreisen immer wieder zu prüfen, ob zum Beispiel ein Gleis frei geworden ist. Mit Lua ist das Unfug und man muss die bisherigen Taktiken überdenken.
Nehmen wir den typischen Fall, dass ein Zug warten muss weil alle Gleise besetzt sind. Dann kann sich doch an dieser Situation nur etwas ändern, wenn ein anderer Zug ein Gleis verlässt. Ich muss also nicht alle Nase lang fragen, ob nun endlich ein Gleis frei geworden ist. Ich muss mir nur merken, dass ein Zug wartet. Der ausfahrende Zug kann dann die Funktion aufrufen, die dem wartenden Zug die Einfahrt ermöglicht.

Wenn man mit einer neuen Anlage beginnt und sich das Skript anschaut, welches automatisch erstellt wird, dann steht in der **EEPMain**() Funktion die Zeile:
**I**=**I**+1
Eine Variable mit dem unleserlichen Namen **I** wird also bei jedem Durchlauf um Eins rauf gezählt.

Damit das klappen kann muss diese Variable außerhalb der Funktion definiert werden. Denn eine Variable,

die nur innerhalb einer Funktion definiert ist, wird bei jedem neuen Funktionsaufruf zurück gesetzt. Deshalb steht ganz am Anfang des Skripts die Zeile
**I**=0

Es ist sehr nützlich die Durchläufe von **EEPMain**() zu zählen. Weil man so einen Takt bekommt, an dem man sich orientieren kann. Man kann zum Beispiel dafür sorgen, dass eine Ampel immer dann umgeschaltet wird, wenn **EEPMain**() 50 Mal durchlaufen wurde.

Ich möchte deshalb anraten dieser Variablen einen lesbaren Namen zu geben.
Schreibt an den Anfang eures Skripts statt
**I**=0
lieber
**TaktGeber**=0

und statt
**I**=**I**+1
lieber
**TatktGeber**=**TaktGeber**+1

Natürlich muss dann auch das **I** in der Zeile
**print**("Counter: ", **I**)
ausgetauscht werden. Also
**print**("Counter: ", **TaktGeber**)

An dem, was das Skript tut, ändert das überhaupt nichts. Es wurde nur der unleserliche Variablenname **I** durch den viel lesbareren Namen **Taktgeber** ersetzt.

Ab dem nächsten Kapitel möchte ich zeigen, was man mit dem bisherigen Wissen in EEP anstellen kann. Dafür möchte ich einen kleinen Schattenbahnhof nutzen, den ich vor längerer Zeit in die Downloadbase gestellt hatte.
Schattenbahnhof in der DLB

Die Anlage enthält eine "klassische" Steuerung mit Schaltautos sowie eine PDF mit Erläuterungen. Schmeißt bitte die beiden Schaltstrecken raus und löscht auch die Zugverbände auf der Anlage. Aber lasst die KPs, welche nicht mit den Schaltstrecken verschwinden, bitte bestehen. Die werden noch benötigt.